Ontology-Based Mental Health Diagnostic System

Patrick Michael

Technische Hochschule Deggendorf

Faculty European Campus Rottal-Inn

GPH-M-2: Elective: Knowledge Based Systems (SS25)

July 14, 2025

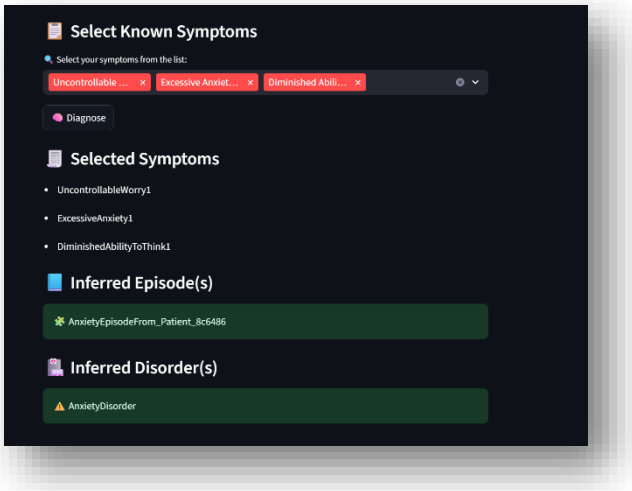
Github Repository

Mental-Health-Ontology

Project Overview

This project aims to use an ontological database, and several Python scripts to create an application that can infer possible mental episodes and disorders from user input of symptoms. It also uses fuzzy matching features to approximate user input and reduce possible errors.

It focuses on diagnosing three categories of mental disorders (major depressive disorder, anxiety disorder, bipolar disorder, manic disorder) via inferring one or more intermediate episodes (depression, anxiety, manic episode, or bipolar episode) from several possible symptoms.



By using the streamlit GUI app to input symptoms, the Python script automatically matches the user input to a static symptom map with different possible alternative naming. It then creates a new patient instance, assigns the symptoms to the created instance via “hasSymptom” object property, and runs the reasoner “HermiT” to infer the possible episodes. It follows by assigning that episode to the created patient instance via “hasEpisode” object property, then saves the newly edited ontology onto a temporary file and reloads it. Finally, it runs the reasoner again to infer possible disorders and shows the diagnosed disorder as an output.

This two-inference chain ensures smooth transition between different symptoms and possible disorders. The modular approach to inference also allows further editing of the ontology and creating more complex structures over time without fear of entanglement of the previous ontology.

Ontology Structure

Class Hierarchy

The ontology is structured around a three-tier class hierarchy.

Symptom Class: which contains several individual clinical signs such as Fatigue, Insomnia, Anhedonia, or Elevated Mood. They are clustered into three superclasses (Depression, Anxiety, Mania) and each symptom has a corresponding instance that can be assigned to patient individuals.

Episode Class: which contains possible intermediate episodes (Anxiety, Bipolar, Depression, Manic) that are inferred from the symptom classes. These also contain corresponding instances to be assigned to the patient instances via the Python diagnostic pipeline.

Disorder Class: which contains the four disorders that can be ultimately inferred via the diagnostic pipeline (Depression, Anxiety, Bipolar, Manic). These can then be displayed as output on the user interface.

Object Properties

These are used to connect various classes to each other and allow the reasoner to infer possible episodes and disorders from the symptom input.

hasSymptom: This is used to connect symptom classes to intermediate episode classes. For example, the class AnxietyEpisode is defined using an existential restriction: (*AnxietyEpisode* ≡ *hasSymptom some (ExcessiveAnxiety or UncontrollableWorry))*. This means that any individual who has at least one symptom that is either ExcessiveAnxiety or UncontrollableWorry can be inferred to have an AnxietyEpisode.

hasEpisode: This is used to connect the episode classes to the final disorder classes. For example, the class AnxietyDisorder is defined using another existential restriction: (*AnxietyDisorder* ≡ *hasEpisode some AnxietyEpisode)*. This means that an individual that has at least one AnxietyEpisode can be inferred to have AnxietyDisorder.

hasDisorder: This can be used to assert the disorders to the patient individuals. It allows further expansion of the ontology.

Technological Components

Ontology File: "PatrickMentalHealthOntologyPython.rdf" that defines the class structure and logical rules followed by the reasoner.

User Interface: *"*app_gui.py" which is built with Streamlit for interactive symptom selection and displaying results.

Reasoning Pipeline: "diagnostic_pipeline.py" that handles patient creation, inference chaining, and ontology reloading.

Symptom Matching: "symptom_map.py" which maps user input to ontology symptom individuals using various predefined variants.

Fuzzy Matching Engine: "fuzzy_matcher.py" that enables flexible input interpretation by using "RapidFuzz".

Dependencies: Listed in "requirements.txt" for easy setup (owlready2, streamlit, RapidFuzz).

Conclusion

This project works as a modular, ontology-based approach to inferring mental health disorders from user reported symptoms. It is a showcase of how semantic technology can support clinical decision-making by combining structured reasoning with a user-friendly interface.

Future improvements can be made by further expanding the ontology to contain more possible symptoms, episodes or disorders, or by adding several datatype properties to understand temporal disease features such as chronic and acute variants. Technological improvements may also include LLM integration to help with semantic analysis from user input and to create a dynamic symptom map that grows and adapts with every subsequent use.